

# IBM Python Assignment

1. Consider a list(list=[]) and perform insertion, print, remove, append, sort, pop, reverse.

**Solution:**

```
list = [1, 2, 4] #Creation of list
```

```
print(list)    #prints the list
```

```
list.insert(2,3) #Inserting interger(3) at position index(2)
```

```
print(list)
```

```
list.remove(1) #Deleting the first element
```

```
print(list)
```

```
list.append(5) #Inserting integer at the end of the list
```

```
print(list)
```

```
list.sort()    #Sorting the list
```

```
print(list)
```

```
list.pop(3)    #Pop the last element of the list
```

```
print(list)
```

```
list.reverse() #Reversing the list
```

```
print(list)
```

### Output:

```
[1, 2, 4]
```

```
[1, 2, 3, 4]
```

```
[2, 3, 4]
```

```
[2, 3, 4, 5]
```

```
[2, 3, 4, 5]
```

```
[2, 3, 4]
```

```
[4, 3, 2]
```

```
main.py
1 list = [1, 2, 4] #Creation of List
2
3 print(list)      #prints the list
4
5 list.insert(2,3) #Inserting interger(3) at position index(2)
6
7 print(list)
8
9 list.remove(1)   #Deleting the first element
10
11 print(list)
12
13 list.append(5)  #Inserting integer at the end of the List
14
15 print(list)
16
17 list.sort()     #Sorting the List
18
19 print(list)
20
21 list.pop(3)     #Pop the Last element of the List
22
23 print(list)
24
25 list.reverse()  #Reversing the List
26
27 print(list)
28
29
```

```
[1, 2, 4]
[1, 2, 3, 4]
[2, 3, 4]
[2, 3, 4, 5]
[2, 3, 4, 5]
[2, 3, 4]
[4, 3, 2]
```

2. Write a calculator program in python.

### Solution:

```
# This function adds two numbers
```

```
def add(x, y):
```

```
    return x + y
```

```
# This function subtracts two numbers
```

```
def subtract(x, y):
```

```
    return x - y
```

```
# This function multiplies two numbers
```

```
def multiply(x, y):
```

```
    return x * y
```

```
# This function divides two numbers
```

```
def divide(x, y):
```

```
    return x / y
```

```
print("Select operation.")
```

```
print("1.Add")
```

```
print("2.Subtract")
```

```
print("3.Multiply")
```

```
print("4.Divide")
```

```
while True:
```

```
    # Take input from the user
```

```
    choice = input("Enter choice(1/2/3/4): ")
```

```
# Check if choice is one of the four options
if choice in ('1', '2', '3', '4'):
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))

    if choice == '1':
        print(num1, "+", num2, "=", add(num1, num2))

    elif choice == '2':
        print(num1, "-", num2, "=", subtract(num1, num2))

    elif choice == '3':
        print(num1, "*", num2, "=", multiply(num1, num2))

    elif choice == '4':
        print(num1, "/", num2, "=", divide(num1, num2))

# Check if user wants another calculation
# Break the while loop if answer is no
calculation = input("Let's do next calculation? (yes/no): ")
if calculation == "no":
    break
else:
    print("Invalid Input")
```

**Output:**

Select operation.

1.Add

2.Subtract

3.Multiply

4.Divide

Enter choice(1/2/3/4): 1

Enter first number: 3

Enter second number: 2

$3.0 + 2.0 = 5.0$

Let's do next calculation? (yes/no): yes

Enter choice(1/2/3/4): 2

Enter first number: 3

Enter second number: 2

$3.0 - 2.0 = 1.0$

Let's do next calculation? (yes/no): yes

Enter choice(1/2/3/4): 3

Enter first number: 3

Enter second number: 2

$3.0 * 2.0 = 6.0$

Let's do next calculation? (yes/no): yes

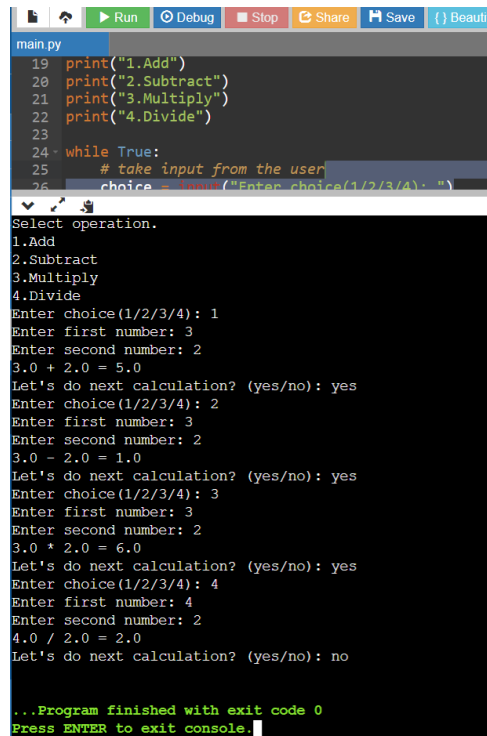
Enter choice(1/2/3/4): 4

Enter first number: 4

Enter second number: 2

$4.0 / 2.0 = 2.0$

Let's do next calculation? (yes/no): no



```
main.py
19 print("1.Add")
20 print("2.Subtract")
21 print("3.Multiply")
22 print("4.Divide")
23
24 while True:
25     # take input from the user
26     choice = input("Enter choice(1/2/3/4): ")

Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4): 1
Enter first number: 3
Enter second number: 2
3.0 + 2.0 = 5.0
Let's do next calculation? (yes/no): yes
Enter choice(1/2/3/4): 2
Enter first number: 3
Enter second number: 2
3.0 - 2.0 = 1.0
Let's do next calculation? (yes/no): yes
Enter choice(1/2/3/4): 3
Enter first number: 3
Enter second number: 2
3.0 * 2.0 = 6.0
Let's do next calculation? (yes/no): yes
Enter choice(1/2/3/4): 4
Enter first number: 4
Enter second number: 2
4.0 / 2.0 = 2.0
Let's do next calculation? (yes/no): no

...Program finished with exit code 0
Press ENTER to exit console.
```

3. Write a program to concatenate, reverse and slice a string.

### Solution:

# Defining strings

```
var1 = "Hello "
```

```
var2 = "World"
```

# + Operator is used to combine strings

```
var3 = var1 + var2
```

```
print(var3)
```

```
txt = "Hello World"[::-1] #Reverse of a string
```

```
print(txt)
```

```
b = "Hello, World!"
```

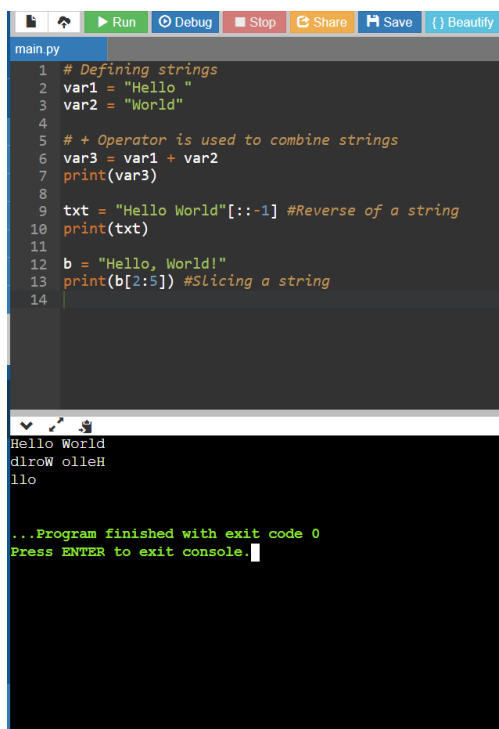
```
print(b[2:5]) #Slicing a string
```

## Output:

Hello World

dlroW olleH

llo

A screenshot of a Python IDE window titled 'main.py'. The code editor shows 14 lines of Python code. Line 1 is a comment: '# Defining strings'. Lines 2-3 define 'var1' as 'Hello ' and 'var2' as 'World'. Line 4 is a comment: '# + Operator is used to combine strings'. Line 5 defines 'var3' as 'var1 + var2'. Line 6 prints 'var3'. Line 7 is a comment: '#Reverse of a string'. Line 8 defines 'txt' as 'Hello World[::-1]'. Line 9 prints 'txt'. Line 10 is a comment: '#Slicing a string'. Line 11 defines 'b' as 'Hello, World!'. Line 12 prints 'b[2:5]'. Line 13 is a comment: '#Slicing a string'. Line 14 is empty. The output console at the bottom shows the results: 'Hello World', 'dlroW olleH', and 'llo'. Below the output, it says '...Program finished with exit code 0' and 'Press ENTER to exit console.'

```
1 # Defining strings
2 var1 = "Hello "
3 var2 = "World"
4
5 # + Operator is used to combine strings
6 var3 = var1 + var2
7 print(var3)
8
9 txt = "Hello World"[::-1] #Reverse of a string
10 print(txt)
11
12 b = "Hello, World!"
13 print(b[2:5]) #Slicing a string
14
```

Hello World  
dlroW olleH  
llo

...Program finished with exit code 0  
Press ENTER to exit console.

## 4. Why is a python a popular programming language?

### Solution:

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
- Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

- Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.
- Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

5. What are the other frameworks that can be used with Python?

**Solution:**

- Django
- Flask
- Dash
- Falcon
- CubicWeb
- CherryPy

6. Full Form of WSGI?

**Solution:**

- WSGI Stands for Web Server Gateway Interface
- WSGI is a specification that describes the communication between web servers and Python web applications or frameworks.
- It explains how a web server communicates with python web applications/frameworks and how web applications/frameworks can be chained for processing a request.